

Docket No. AUS920030471US1

**METHOD AND APPARATUS FOR TRANSFERRING DATA FROM A
MEMORY SUBSYSTEM TO A NETWORK ADAPTER FOR IMPROVING THE
MEMORY SUBSYSTEM AND PCI BUS EFFICIENCY**

5 **CROSS REFERENCE TO RELATED APPLICATIONS**

The present invention is related to an application
entitled "Method and Apparatus for Transferring Data From
a Memory Subsystem to a Network Adapter by Extending Data
10 Lengths to Improve the Memory Subsystem and PCI Bus
Efficiency", serial no. _____, attorney docket no.
AUS920030470US1, filed even date hereof, assigned to the
same assignee, and incorporated herein by reference.

15 **BACKGROUND OF THE INVENTION**

1. Technical Field:

The present invention relates generally to an
improved data processing system and in particular to a
20 method and apparatus for transferring data. Still more
particularly, the present invention relates to a method
and apparatus for transferring data from a memory
subsystem to a network adapter.

25 **2. Description of Related Art:**

The data processing systems includes a bus
architecture for transferring data between various
components. One type of bus architecture is a Peripheral
Component Interconnect (PCI). PCI provides a high-speed

Docket No. AUS920030471US1

data path between the CPU and peripheral devices, such as memory subsystem, a network adapter, and a video adapter.

With respect to transferring data between a memory subsystem and an input/output (I/O) subsystem using a PCI bus, efficiencies in transferring data are dependent on
5 cache aligned data transfers from the memory subsystem to the I/O subsystem. Efficiencies are greatest when the total data transfer is an integral multiple of the cache line size (CLS). For example, transfers to a disk
10 storage system fit this model in which typical transfers have sizes, such as 512, 1024, 2048, and 4096 bytes.

These efficiencies are typically not found with some I/O subsystems, such as network adapters. For example, a maximum Ethernet frame size is 1514 bytes, which is not
15 divisible by any CLS. A CLS is typically 2^n in size. As a result, the remainder of the data is transferred in a small quantity, requiring I/O cycles. This type of overhead becomes significant for high bandwidth network adapters, such as those capable of transferring 10 Gbs.

20 Therefore, it would be advantageous to have an improved method, apparatus, and computer instructions for transferring data from a memory to a network adapter.

Docket No. AUS920030471US1

SUMMARY OF THE INVENTION

The present invention provides a method, apparatus,
5 and computer instructions for transferring data from a
memory to a network adapter. A request is received to
transfer data to a network adapter. An offset is set for
a starting address of the data to align the data with an
end of a frame in the memory, wherein the frame is
10 transferred from the memory to the network adapter.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a pictorial representation of a data processing system in which the present invention may be implemented in accordance with a preferred embodiment of the present invention;

Figure 2 is a block diagram of a data processing system is shown in which the present invention may be implemented;

Figure 3 is a diagram illustrating components used in transferring data from a memory subsystem to a network adapter in accordance with a preferred embodiment of the present invention;

Figure 4 is a diagram illustrating a table of data transfer commands made in transferring data from a memory subsystem to a network adapter using a current transfer process;

Figure 5 is a diagram illustrating a frame buffer in accordance with a preferred embodiment of the present invention;

Docket No. AUS920030471US1

Figure 6 is a diagram of commands used to transfer a frame of data in accordance with a preferred embodiment of the present invention; and

Figure 7 is a flowchart of a process for aligning
5 data with an offset to maximize transfer efficiency in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures and in particular
5 with reference to **Figure 1**, a pictorial representation of
a data processing system in which the present invention
may be implemented is depicted in accordance with a
preferred embodiment of the present invention. A
computer **100** is depicted which includes system unit **102**,
10 video display terminal **104**, keyboard **106**, storage devices
108, which may include floppy drives and other types of
permanent and removable storage media, and mouse **110**.
Additional input devices may be included with personal
computer **100**, such as, for example, a joystick, touchpad,
15 touch screen, trackball, microphone, and the like.
Computer **100** can be implemented using any suitable
computer, such as an IBM eServer computer or
IntelliStation computer, which are products of
International Business Machines Corporation, located in
20 Armonk, New York. Although the depicted representation
shows a computer, other embodiments of the present
invention may be implemented in other types of data
processing systems, such as a network computer. Computer
100 also preferably includes a graphical user interface
25 (GUI) that may be implemented by means of systems
software residing in computer readable media in operation
within computer **100**.

With reference now to **Figure 2**, a block diagram of a
data processing system is shown in which the present
30 invention may be implemented. Data processing system **200**

Docket No. AUS920030471US1

is an example of a computer, such as computer 100 in Figure 1, in which code or instructions implementing the processes of the present invention may be located. Data processing system 200 employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor 202 and main memory 204 are connected to PCI local bus 206 through PCI bridge 208. PCI bridge 208 also may include an integrated memory controller and cache memory for processor 202. Additional connections to PCI local bus 206 may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter 210, small computer system interface SCSI host bus adapter 212, and expansion bus interface 214 are connected to PCI local bus 206 by direct component connection. In contrast, audio adapter 216, graphics adapter 218, and audio/video adapter 219 are connected to PCI local bus 206 by add-in boards inserted into expansion slots. Expansion bus interface 214 provides a connection for a keyboard and mouse adapter 220, modem 222, and additional memory 224. SCSI host bus adapter 212 provides a connection for hard disk drive 226, tape drive 228, and CD-ROM drive 230. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor 202 and is used to coordinate and provide control of various components within data processing system 200 in Figure 2. The

Docket No. AUS920030471US1

operating system may be a commercially available operating system such as Windows XP, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the
5 operating system and provides calls to the operating system from Java programs or applications executing on data processing system 200. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented programming system, and applications
10 or programs are located on storage devices, such as hard disk drive 226, and may be loaded into main memory 204 for execution by processor 202.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 2** may vary depending on the
15 implementation. Other internal hardware or peripheral devices, such as flash read-only memory (ROM), equivalent nonvolatile memory, or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 2**. Also, the processes of the present
20 invention may be applied to a multiprocessor data processing system.

For example, data processing system 200, if optionally configured as a network computer, may not include SCSI host bus adapter 212, hard disk drive 226,
25 tape drive 228, and CD-ROM 230. In that case, the computer, to be properly called a client computer, includes some type of network communication interface, such as LAN adapter 210, modem 222, or the like. As another example, data processing system 200 may be a
30 stand-alone system configured to be bootable without

Docket No. AUS920030471US1

relying on some type of network communication interface, whether or not data processing system 200 comprises some type of network communication interface. As a further example, data processing system 200 may be a personal
5 digital assistant (PDA), which is configured with ROM and/or flash ROM to provide non-volatile memory for storing operating system files and/or user-generated data.

The depicted example in **Figure 2** and above-described
10 examples are not meant to imply architectural limitations. For example, data processing system 200 also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system 200 also may be a kiosk or a Web appliance.

15 The processes of the present invention are performed by processor 202 using computer implemented instructions, which may be located in a memory such as, for example, main memory 204, memory 224, or in one or more peripheral devices 226-230.

20 Turning now to **Figure 3**, a diagram illustrating components used in transferring data from a memory subsystem to a network adapter is depicted in accordance with a preferred embodiment of the present invention. The components illustrated in **Figure 3** may be implemented
25 in a data processing system, such as data processing system 200 in **Figure 2**.

In this example, data is transferred from memory subsystem 300 to network adapter 302 using I/O bridge 304. Memory subsystem 300 includes memory 306 and memory
30 controller 308. In this example, memory 306 is a system

Docket No. AUS920030471US1

or main memory, such as main memory 204 in **Figure 2**.

Memory controller 308 may take various forms, such as a processor or an application specific integrated circuit (ASIC). The data to be transferred is located in memory

5 306. Memory controller 308 is used to control the transfer of data from memory 306 to I/O bridge 304, which may be found in host/PCI/cache bridge 208 in **Figure 2**. This I/O bridge interfaces with the processor and the memory on one side, and provides an interface to the PCI
10 bus on the other side.

In this example, I/O bridge 304 includes prefetch memory 310 and non-prefetch memory 312. Access to these memories and the transfer of the data using these memories is handled by control logic 314. Control logic
15 314 may be implemented in different forms, such as a processor or an ASIC. The present invention provides improved cache aligned memory read operations by modifying the starting address for the data transfer in the system memory, memory 306. This modification is
20 performed in system memory 306 in these examples.

Cache aligned memory read operations occur using prefetch memory 310. Memory read operations that are not cache aligned occur using non-prefetch memory 312. When a PCI command, MR or MRL is issued, the data is fetched
25 from the system memory, memory 306, into non-prefetch memory 312, whereas if a MRM command is issued, the data is fetched into prefetch memory 310. These memories are high speed arrays to match with the PCI bus speeds. Typically, non-prefetch memory is a cache line size (128
30 bytes), and prefetch memory is of multiple cache lines

Docket No. AUS920030471US1

(n * cache line). These examples are implementation specific and can vary from system to system. The memory buffers are located in the I/O bridge.

Network adapter 302 reads data from I/O bridge 304
5 to generate and send frame 316 onto network 318 in these examples. If the memory is cache aligned, the data may be read from prefetch memory 310.

Turning now to **Figure 4**, a diagram illustrating a table of data transfer commands made in transferring data
10 from a memory subsystem to a network adapter using a current transfer process is depicted. With a PCI architecture, three classes of memory reads are present. These memory reads are memory read (MR), memory read line (MRL), and memory read multiple (MRM). A MR command is
15 used to read 1 to 8 bytes of data from a non-prefetchable memory in a single PCI cycle, such as one address phase and one data phase. A MRL command is used to read more than a double word up to the next cache line boundary in a prefetchable address space. MRM is a command used to
20 read a block of data, which crosses a cache line boundary of data in a prefetchable address space.

In table 400, a series of commands are illustrated to show how these different types of commands are used in a typical Ethernet frame having a size of 1514 bytes in
25 which this data is fetched by network adapter from system memory. In this example, the CLS is assumed to be 128 bytes. The I/O bridge in this example has a prefetch capability of 512 bytes with a PCI bus having a width of 64 bits (8 bytes).

Docket No. AUS920030471US1

Entries **402-434** contain commands used to transfer a 1514 byte Ethernet frame from system memory to the network adapter. Entries **402-406** employ MRM commands used to transfer 1408 bytes of data. Entries **408-434**
5 contain MR commands used to transfer the remaining bytes of data needed to form a 1514 byte frame. As can be seen, all of these MR commands waste bandwidth on the memory sub-system and I/O bus.

Turning next to **Figure 5**, a diagram illustrating a
10 frame buffer is depicted in accordance with a preferred embodiment of the present invention. In this example, frame buffer **500** is an example of a frame buffer in a system memory, such as memory **306** in **Figure 3**. In this example, no data is present in section **502** of frame
15 buffer **500**. Next, 490 bytes of data are present in section **504**, 512 bytes of data are present in section **506**, and 512 bytes of data are present in section **508** of frame buffer **500**.

Typically, data in the frame buffer is aligned with
20 the beginning of frame buffer **500** at address **508**. In these examples, data is offset to address **510**. This offset is the start of an Ethernet frame. This offset is made within frame buffer **500** to align the end of the frame with the last transfer of a cache line at the end
25 of frame buffer **500**. This end of frame cache alignment is present at address **512**. The end of frame cache alignment allows more efficient transfer of data in contrast to having the frame start at the beginning of the frame buffer because the adapter issues MRL or MRM
30 command if the amount of data equals or exceeds cache

Docket No. AUS920030471US1

line. By introducing an offset in the beginning, the last transfer always ends with a MRL or MRM command instead of MR command.

In these examples, the Ethernet frame in frame buffer 500 may have a size ranging from 64 bytes to 1514 bytes. The PCI architecture allows for a CLS in powers of 2, such as 64, 128, and 256. An offset for frame buffer 500 may be set to zero if the frame size is divisible by the CLS without a remainder. If the frame size is not divisible by the CLS without a remainder, then the frame buffer offset is equal to the CLS minus the remainder bytes. The remainder bytes are equal to the frame size minus the total cache aligned bytes. The total cache aligned bytes is found using the following equation:

$$\text{total cache aligned bytes} = \text{ABS}(\text{frame size} / \text{CLS}) * \text{CLS}.$$

Turning next to **Figure 6**, a diagram of commands used to transfer a frame of data is depicted in accordance with a preferred embodiment of the present invention. When the end of the frame is aligned with the last transfer of a cache line as illustrated above in **Figure 5**, a smaller number of commands are needed to transfer the data. In this example, table 600 includes only three entries, 602, 604, and 606. In this example, only MRM commands are used. MR commands are no longer required in the transfer of this data when the offset is used. As can be seen, in contrast to the number of commands used in table 400 in **Figure 4**, a much smaller number of commands are needed to transfer the same amount of data for a 1514 byte Ethernet frame.

Docket No. AUS920030471US1

Turning next to **Figure 7**, a flowchart of a process for aligning data with an offset to maximize transfer efficiency is depicted in accordance with a preferred embodiment of the present invention. The process
5 illustrated in **Figure 7** may be implemented in components such as memory controller 308 and control logic 314 in **Figure 3**.

The process begins by receiving a request to transfer data to a network adapter (step 700).
10 Thereafter, a frame size is identified (step 702), and CLS is obtained (step 704). CLS is a static value for a given system and is not dynamically changed. This value is initialized by the system firmware during power up sequence. Next, a determination is made as to whether
15 the frame size is divisible by the CLS without a remainder (step 706).

If the frame size is not divisible by the CLS without a remainder, then the frame buffer offset is set equal to the CLS minus the remainder bytes (RB) (step
20 708). The RB is equal to the frame size minus the total cache aligned bytes (TCAB). This is identified from the following equation:

$$TCAB = \text{ABS}(\text{frame size} / \text{CLS}) * \text{CLS}.$$

The identified frame buffer offset is used to offset
25 data in the buffer (step 710). This offset is used to provide for an alignment for the end of the frame with the cache line. Next, the transfer of data to the network adapter is initiated (step 712) with the process terminating thereafter.

Docket No. AUS920030471US1

With reference again to step 706, if the frame size is not divisible by the CLS without a remainder, the frame buffer offset is equal to zero (step 714). The process then proceeds to step 710 as described above.

5 In this manner, the present invention provides a method, apparatus, and computer instructions for transferring data from a memory to a network adapter. The mechanism of the present invention allows for a significant reduction in the number of PCI bus commands
10 needed to transfer data to a network adapter across a PCI bus. The mechanism of the present invention achieves the efficiencies by modifying the starting address of the data transfer in the memory. This offset is selected to have the data at the end of the frame aligned with the
15 last transfer of the cache line.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of
20 the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the
25 distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications
30 links using transmission forms, such as, for example,

Docket No. AUS920030471US1

radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

5 The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in
10 the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are
15 suited to the particular use contemplated.